METHOD AND SYSTEM FOR INTERNET TRANSPORT ACCELERATION WITHOUT PROTOCOL OFFLOAD

BACKGROUND OF THE INVENTION

Field of the Invention

[01] The present invention relates to a network system. More particularly, the present invention relates to a system and a method for direct data placement of data for an application that uses a network protocol, such as TCP/IP, SNA and/or IPX

Description of the Related Art

[02] Internet protocols, such as TCP/IP, and other networking protocols, such as SNA and IPX, are crucial to the performance of many applications. The vast majority of communication protocols for current applications use TCP/IP as the transport protocol. Consequently, both researchers and practitioners have focused on performance improvements for TCP/IP.

[03] The most popular conventional approach for improving TCP/IP performance is to offload the entire TCP/IP stack onto a network adapter. See, for example, U.S. Patent No. 6,434,620 B1 to L.B. Boucher et al. and U.S. Patent Application 2002/016191 A1 to L.B. Boucher et al. Offloading the entire TCP/IP stack onto a network adapter has many advantages. For example, interrupt overhead for the host processor is reduced because there is only one interrupt per TCP message as opposed to one interrupt for each TCP segment. Offloading the entire TCP/IP stack also relieves the burden of TCP/IP processing from the host processor, particularly the copy-and-checksum processing overheads, and saves precious CPU cycles. Further, offloading the TCP/IP stack provides the opportunity for direct data placement on the receive path based on application header processing on the network adapter.

[04] A drawback of offloading the TCP/IP stack onto a network adapter, however, is that the processing power of a network adapter is not comparable to the processing power of a

general purpose CPU and such offloading may cause bottlenecks. Recent studies have shown that the performance of the TCP/IP offload adapters is sometimes behind that of a software TCP/IP stack.

[05] Consequently, what is needed is a technique for direct data placement of data for a TCP/IP application without offloading the protocol onto a network adapter.

BRIEF SUMMARY OF THE INVENTION

[06] The present invention provides a technique for direct data placement of data for a TCP/IP application without offloading the protocol onto a network adapter.

[07] The advantages of the present invention are provided by a system and method for direct data placement of data for an application that uses a network protocol, such as TCP/IP, SNA or IPX. According to the invention, an application packet header is detected using a packet classifier within a network adapter. The application packet header belongs to a packet in a data stream associated with the application. Offsets included within the application header are then identified and a plurality of registers is loaded with the identified offsets. Each of a plurality direct data placement patterns are masked with contents of the loaded registers. Direct data placement of data associated with the application packet header is initiated when a result of masking a set of values corresponding to a direct data placement pattern with contents of the loaded registers matches one of at least one direct data placement pattern. Each direct data placement pattern is associated with an application packet header and includes a corresponding I/O context. Additionally, each direct data placement pattern includes a corresponding connection path for direct placement of a payload corresponding to the detected application header in a memory that is utilized by the application.

[08] When direct data placement is initiated, information corresponding to the detected application header is extracted and the data payload of the detected applications header is DMA-ed to

registered memory that is associated with the application, based on the direct data placement pattern. Only one interrupt of a host processor for the network adapter is generated for each message. Moreover, when direct data placement of data associated with the application packet header is initiated, a host processor for the network adapter does not perform copy and checksum processing.

BRIEF DESCRIPTION OF THE DRAWINGS

[09] The present invention is illustrated by way of example and not by limitation in the accompanying figures in which like reference numerals indicate similar elements and in which:

[10] Figure 1 shows a functional block diagram of a computer system having a network adapter with a packet classifier according to the present invention; and

[11] Figure 2 show a flowchart of a process for accelerating TCP/IP applications without offloading the TCP/IP stack onto a network adapter according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[12] The present invention provides a technique for accelerating TCP/IP applications without offloading the TCP/IP stack onto a network adapter. Instead, the present invention offloads application-programmable intelligence onto a network adapter that provides a programmable packet classifier for direct data placement for TCP applications, thereby providing most of the benefits associated with conventional offloading of the TCP/IP stack to a network adapter.

[13] The present invention provides many advantages. For example, system interrupts are reduced because the system is interrupted only on the transfer of data and not for every incoming TCP/IP packet. Additionally, TCP/IP copy overhead is eliminated because the programmable interface does direct data placement based on the classifier recognizing application headers. Moreover, direct data placement is an inbuilt advantage of the present invention.

[14] While the present invention is as advantageous as a conventional system that offloads TCP/IP stack processing, the present invention provides two additional benefits over such conventional systems. First, the complexity of the programmable packet classifier of the present invention is much less than the complexity of the entire TCP/IP stack. Consequently, a network adapter having a programmable packet classifier of the present invention is less costly. Second, the present invention leaves TCP/IP processing on the host processor, thereby allowing the performance to scale with CPU speeds.

[15] Figure 1 shows a functional block diagram of a computer system 100 having a network adapter with a packet classifier according to the present invention. Computer system 100 includes a host processor 101, a TCP/IP stack 102 and a network adapter 103. Network adapter 103 includes a packet classifier 104 having a programming interface 105, and is connected to network 106, such as an Ethernet.

[16] Programming interface 105 for packet classifier 104 includes a programmable interface that is programmed by a TCP/IP application. The first programmable parameter is a set of registers R that contain values that are to be loaded from a TCP stream. The second programmable parameter is the set of offsets O that indicate where the values loaded into registers R are located in the application header. The third programmable parameter is a set of masks M that are applied to the contents of registers R. The fourth programmable parameter is a set of values V that are to be matched with the results of masking the contents of registers R masked with the contents of masks M. The fifth programmable parameter is the action A that is to be taken when a pattern is matched. Action A can have additional associated parameters depending on the action taken. The application separately programs a cache of I/O tags to reserved memory RM for each corresponding I/O tag in a well-known manner.

[17] The present invention requires that the TCP stream has application header synchronization. Additionally, the present invention does not provide application header resynchronization

because loss of application header synchronization is usually handled by protocol action, such as by markers, or by breaking down the connection.

[18] Figure 2 show a flowchart 200 of a process for accelerating TCP/IP applications without offloading the TCP/IP stack onto a network adapter according to the present invention. At step 201, a TCP/IP application programs the R, O, M, V and A programmable parameters of packet classifier 104. At step 202, TCP/IP communications are monitor by packet classifier 104 and it is determined whether an application header is detected. If not, flow remains at step 202. Otherwise, when packet classifier detects an application header, flow continues to step 203 where packet classifier 104 loads registers R from offsets O in the detected header and at step 204 masks values V with the contents of registers R. At step 205, it is determined whether the results of the masking match the results with a corresponding programmed pattern. When the results of the masking match a programmed patterned, flow continues to step 206 where packet classifier 104 takes action A that is specified in the pattern. When action A involves direct data placement, head information (i.e., I/O tags and protocol details) are extracted for DMA-ing the data payload of the of the application header to reserved memory RM associated with the application. For example, when an I/O tag corresponds to reserved memory RM1-RM2, action A involves moving the contents of TCP sequence numbers S1-S2 to RM1-RM2. Flow continues to step 202.

[19] If, at step 205, the results of the masking do not match a programmed pattern, then flow continues to step 207 where it is determined whether there are no TCP header in incoming packets because there is IP fragmentation. If so, then flow continues to step 208 where the task of moving the contents of TCP sequence numbers is hindered by the fragmentation and is best done through action A' in which additional programming of registers R', masks M', values V' and offsets O' is done to map the fragmented IP headers to the TCP stream. Flow continues to step 202.

[20] If at step 207, IP fragmentation is not detected, flow continues to step 209 where the incoming TCP/IP packets doe not match any pattern because there are out-of-order packets causing missing application headers. At step 209, the incoming TCP/IP packets are sent directly to the host TCP/IP stack for processing. At this point, the TCP/IP packet is subjected to the copy-and-checksum overhead. Alternatively, the out-of-order packets are buffered in network adapter 103. For high data rates, though, this alternative increases the memory requirements of network adapter 103.

[21] While the present invention has been described using TCP/IP as an exemplary network protocol, it should be understood that the present invention is applicable for other network protocols, such as SNA or IPX.

[22] Although the foregoing invention has been described in some detail for purposes of clarity of understanding, it will be apparent that certain changes and modifications may be practiced that are within the scope of the appended claims. Accordingly, the present embodiments are to be considered as illustrative and not restrictive, and the invention is not to be limited to the details given herein, but may be modified within the scope and equivalents of the appended claims.